Towards Network-accelerated ML-based Distributed Computer Vision Systems

Hisham Siddique
Miguel Neves
Carson Kuzniar
Israat Haque

DALHOUSIE UNIVERSITY

IEEE ICPADS

PINet Programmable and Intelligent Networking

Presentation Outline

- Context and Motivation
- NetPixel
- System Design
- Evaluation
- Conclusion and Future Work

Context and Motivation

- IoT: Internet of Things
- Latency-critical applications have seen a resurgence
  - Augmented Reality/Virtual Reality
  - Intelligent transport systems
  - Drones/Surveillance
- The cloud cannot service these applications at the required latency requirements
- *Edge computing*: offers computational resources nearby to end-devices (at the network edge)

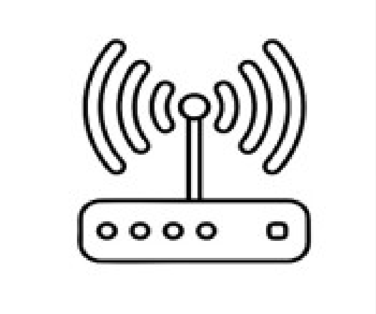
Context and Motivation

- Programmable Networking devices (PNDs)
  - Located at low hop count
  - High packet processing capabilities

- NetPixel targets these PNDs
  - Based on decision tree
  - Simple implementation
  - Competitive accuracy

DT-based model

One-switch architecture
Simple, highly interpretable



P4: Programming the Data Plane

- The P4 language allows programmability of PNDs
- Protocol and target independent
- Makes use of the PISA architecture
  - Pipeline Forwarding Architecture

- Contains efficient stacks
  - Match-action Tables
  - Stateful elements

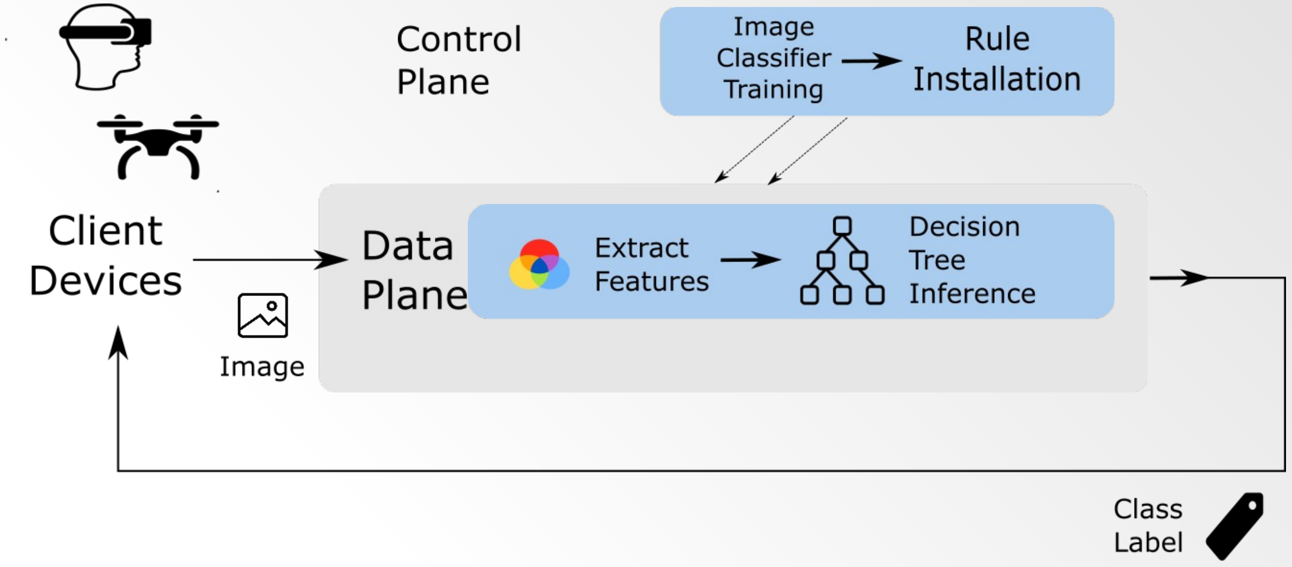
System Overview

Switch

- Convert RGB values to grayscale
- Extract features from each chunk packet
- Store values on switch register
- Apply decision tree after all chunks received
- Send back class label

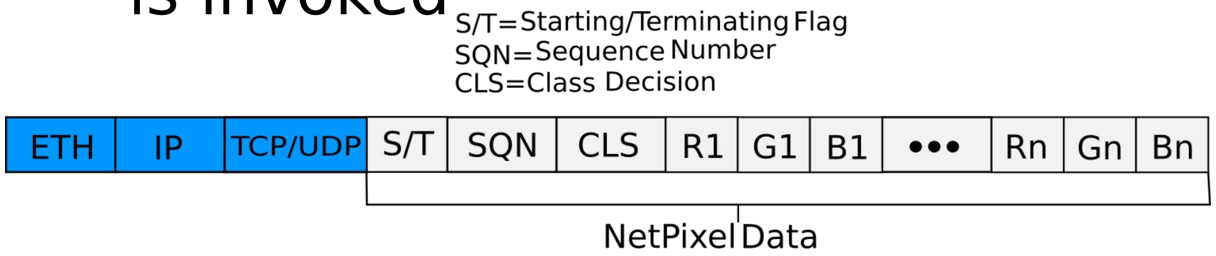
Controller

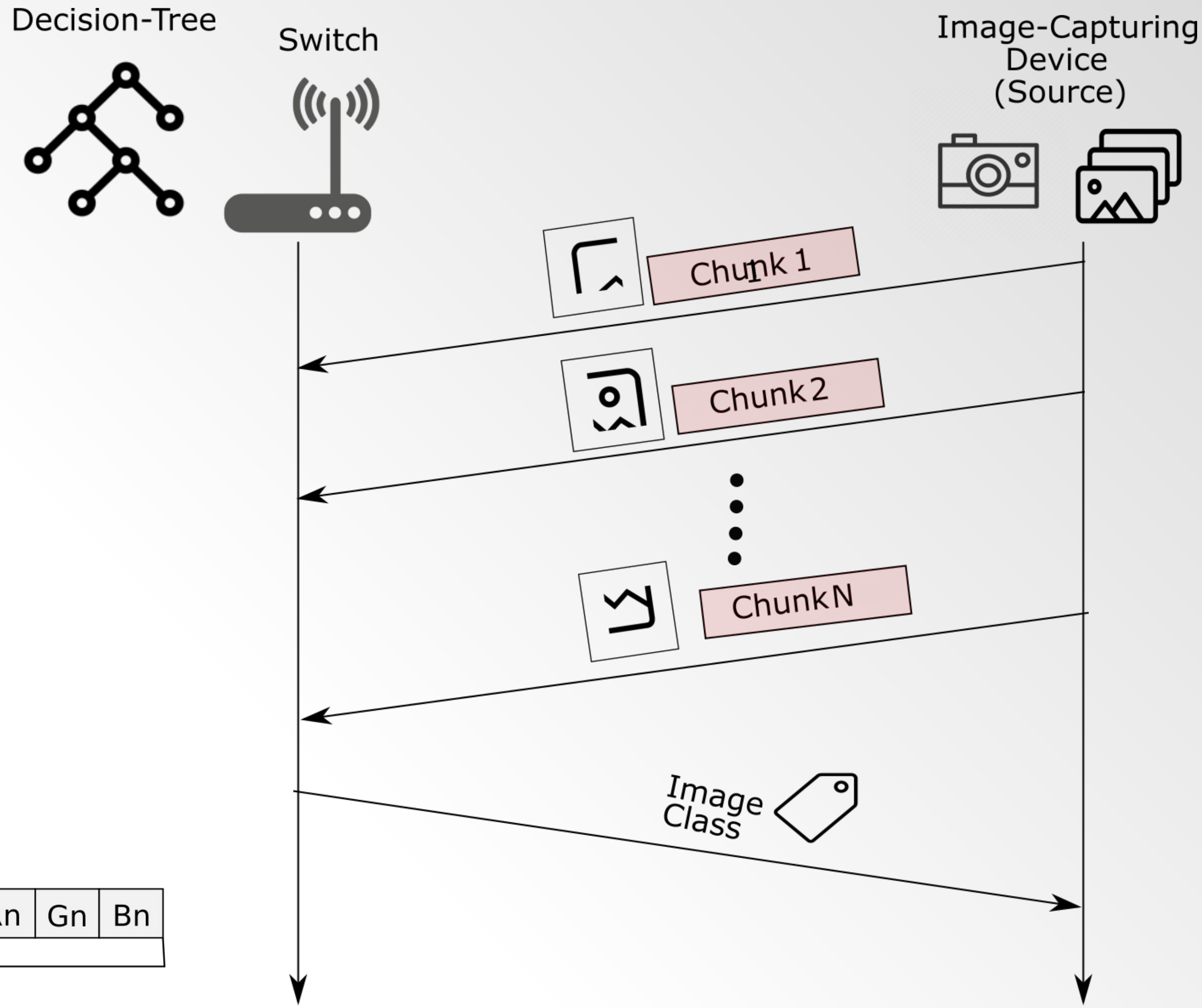
- Train Decision Tree
- Install rules on switch



NetPixel Protocol

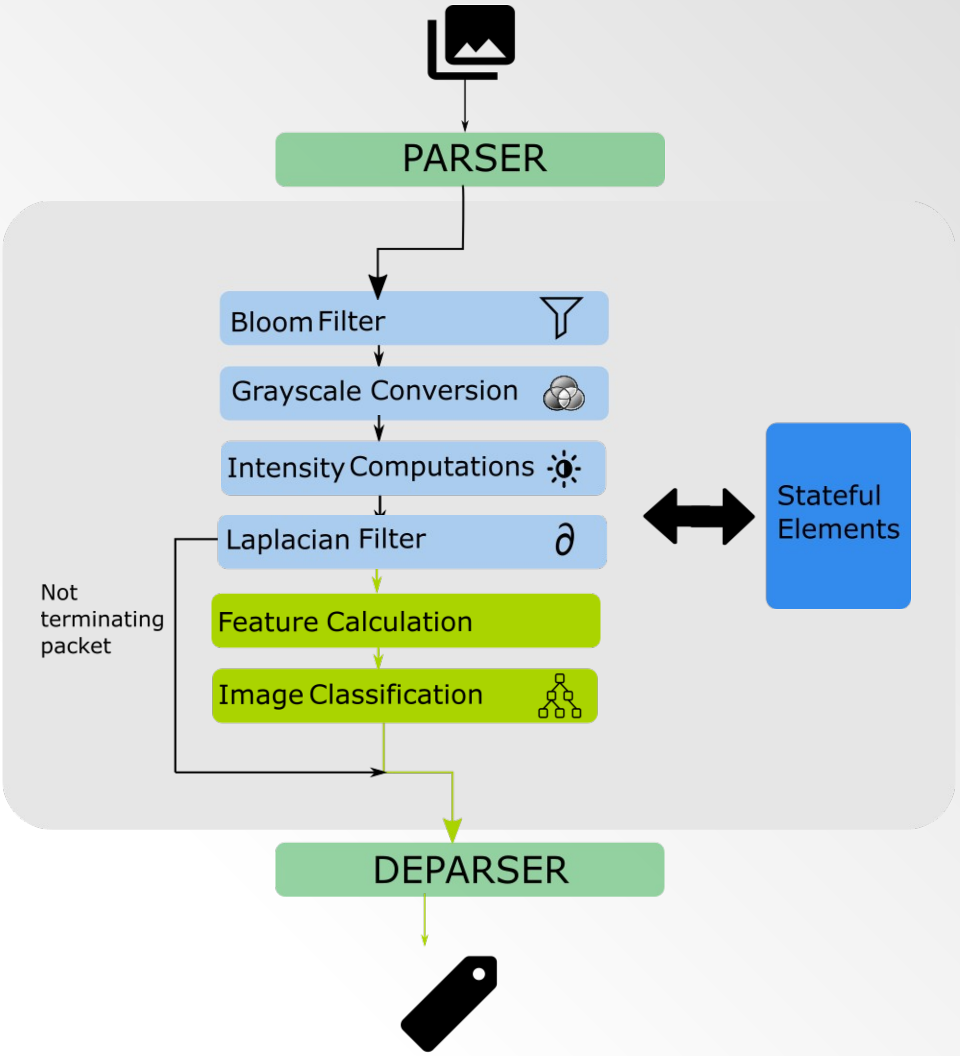
- Images are sent over the network as subsequent chunks
- Each packet contains a chunk as the RGB pixel values
- Once all chunks are received, the decision tree is invoked





NetPixel Pipeline

- Packets are parsed to retrieve chunk information
- Converted to grayscale values after Bloom filter
- Remaining features calculated
- If not a terminating packet, discarded
- Else, features consolidated and image labelled



Evaluation Setup

Setup

- Target model: Bmv2
- Architecture: v1Model
- Hosts emulated using python scapy library

Baseline system

- Decision Tree implemented using Python
- Trained using the CART algorithm
- Not constrained by architecture

Dataset	Image size	Training images	# of labels
MNIST	28x28x1	60000	10
CalTech101	Variable	9200	101
CalTech256	Variable	30000	256
ImageNet	Variable	20000	100

GitHub

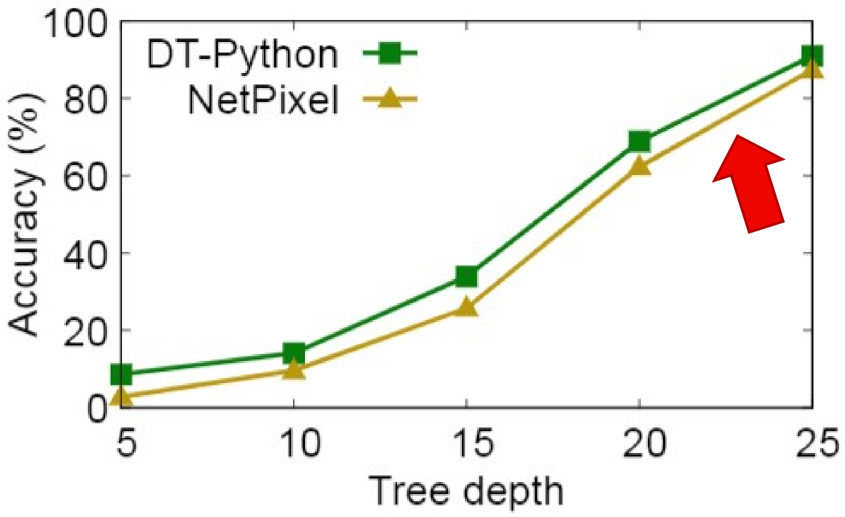
[PINetDalhousie/netpixel (github.com)](https://github.com/PINetDalhousie/netpixel)

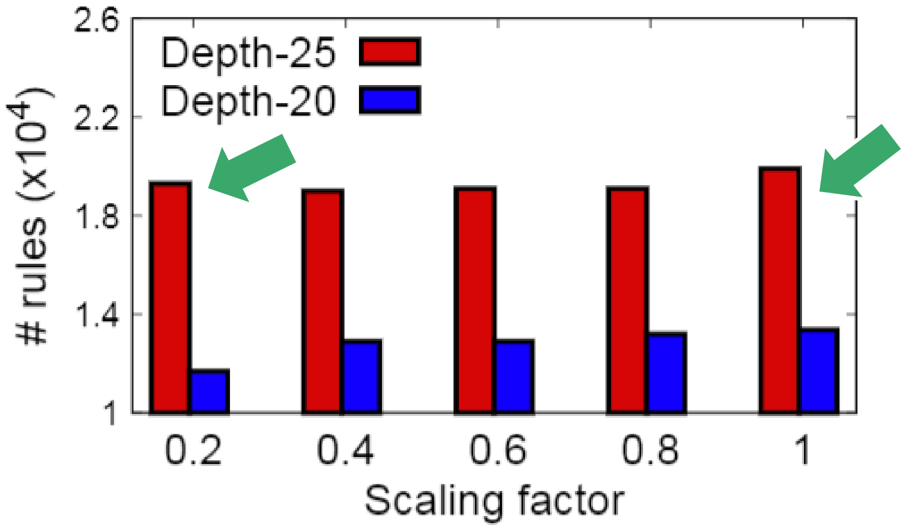
Results

Dataset	DT-Python	NetPixel
MNIST	92.45%	85.00%
CalTech101	96.50%	92.78%
CalTech256	91.11%	87.28%
ImageNet	90.36%	86.73%

- A maximum discrepancy of 7.45% was observed
- CalTech datasets performed well due to a varied imageset
- MNIST performed worse comparatively due to images being largely similar

Evaluation and Results





- Impact of tree depth and image scales were also evaluated
  - Caltech256 dataset used
- Accuracy shows a steady increase as tree depth is increased
- Scaling factor does not have a significant effect on

Conclusions

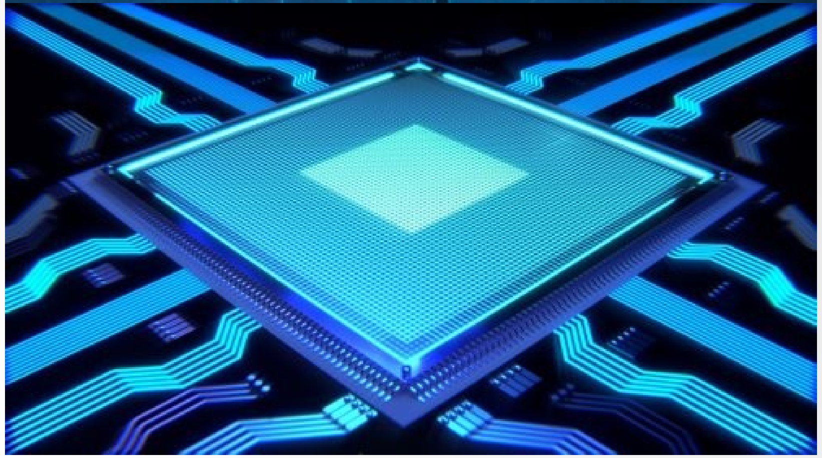
- Proposed a novel system that allows accurate image classification on a programmable network device for latency-critical applications.

- Uses a protocol for sending images as chunks through multiple packets to allow efficient calculation of features or feature maps

- The decision tree based system was evaluated against a baseline server implementation and demonstrated its competitive accuracy for different image datasets

Future Research

A number of extensions are possible for this work



- Deployment on hardware
  - Tofino switch
  - Can evaluate latency gains
- Extension to other image related tasks
- Classification of non-image data

Thank You